Intelligent Systems

Dijkstra & Prim

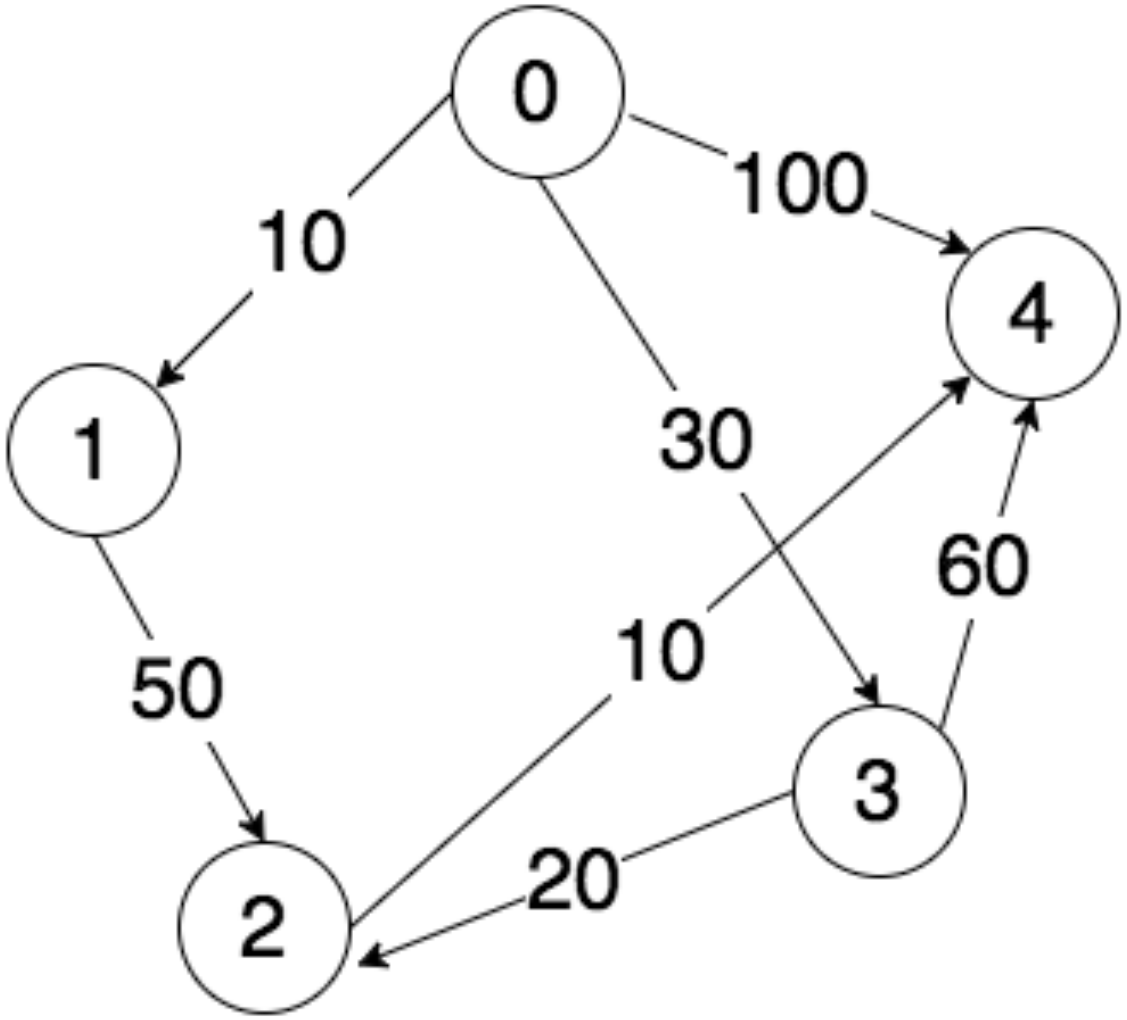
Sinan NAR

Content

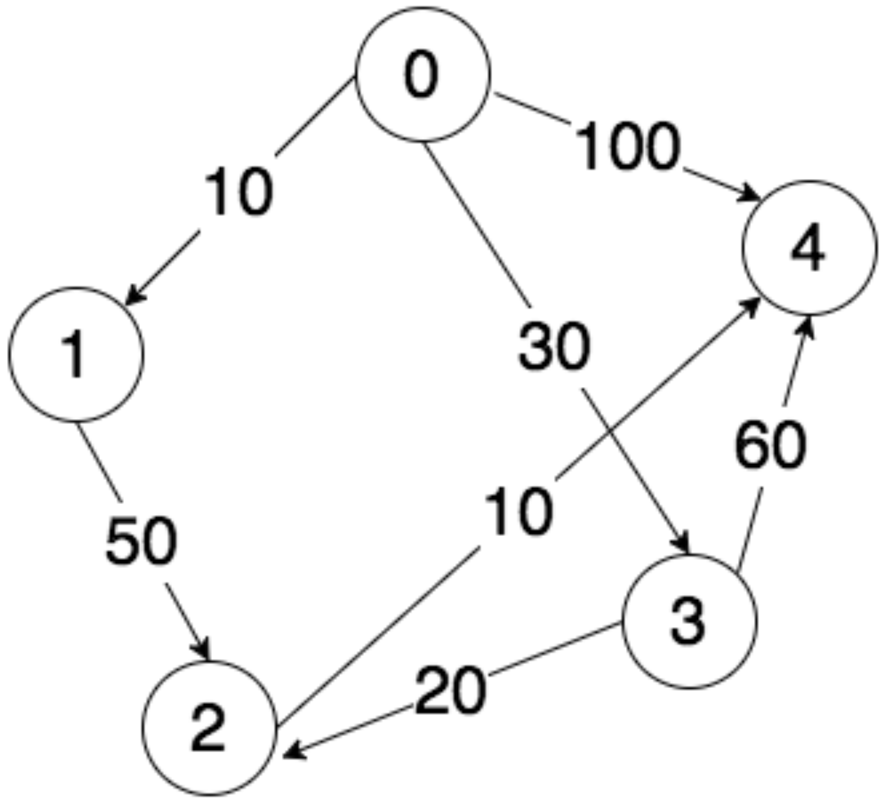
- Dijkstra - How to pronounce & Shortest Path
- Prim - Minimum Spanning Tree
- Implementation (I will not explain code)
- Algorithm Details

DIJKSTRA

- Shortest path from one vertex to other vertices
- Need two sets, S and V-S
- Will be using tables p[v] and d[v]



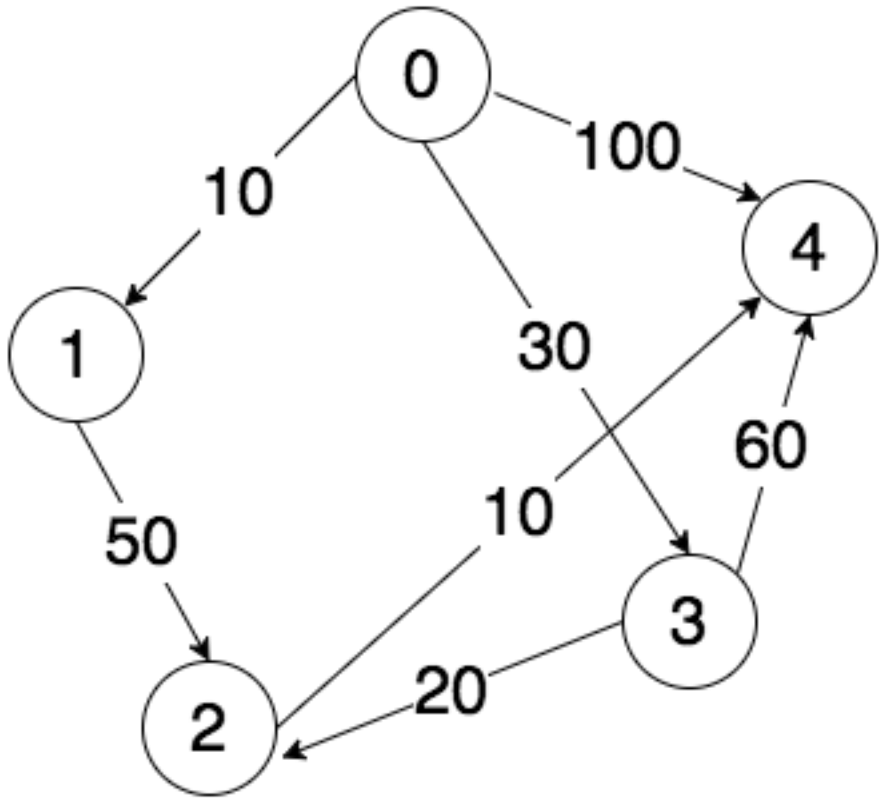
STEP 1



S = {0}
V-S = {1,2,3,4}

V	d[V]	p[V]	
1	10	0	★
2	inf	0	
3	30	0	
4	100	0	

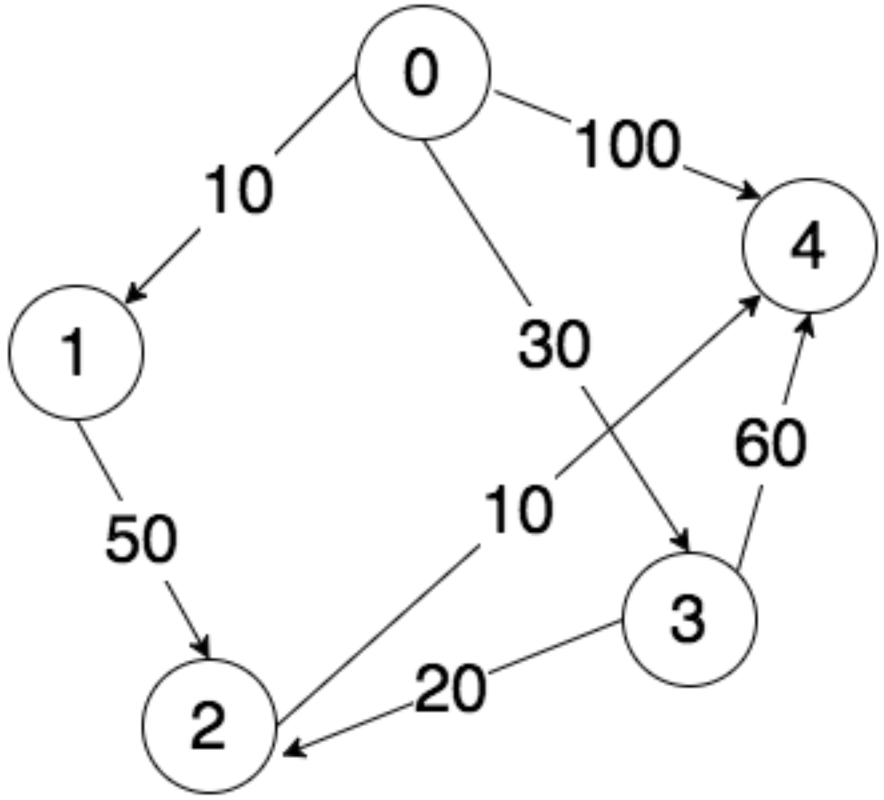
STEP 2



S = {0,1}
V-S = {2,3,4}

V	d[V]	p[V]	
1	10	0	
2	60	1	
3	30	0	★
4	100	0	

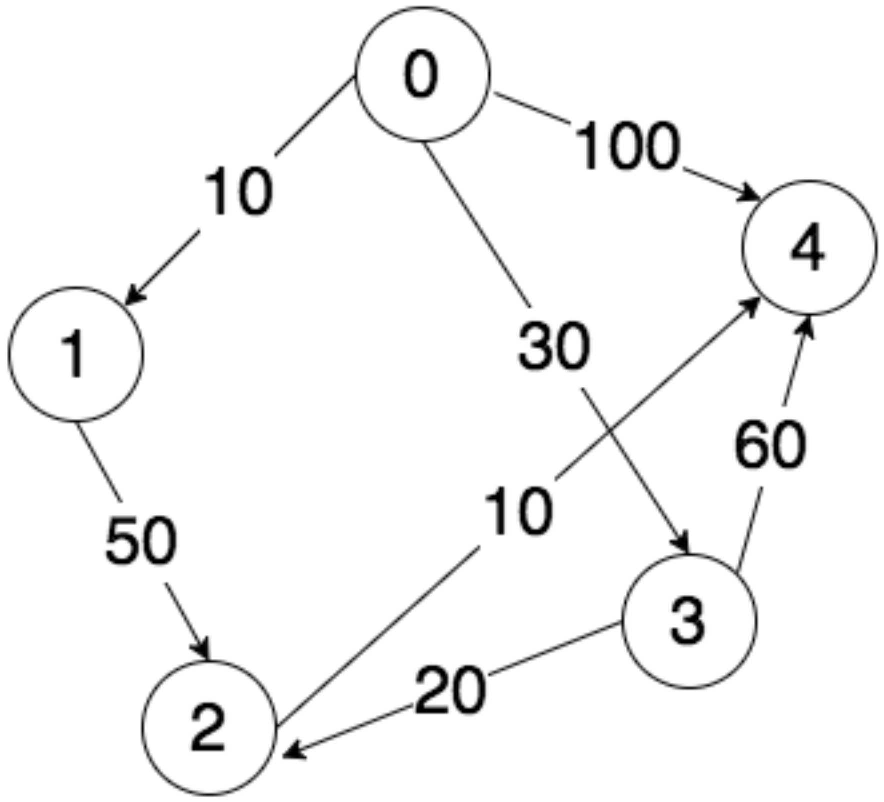
STEP 3



S = {0,1,3}
V-S = {2,4}

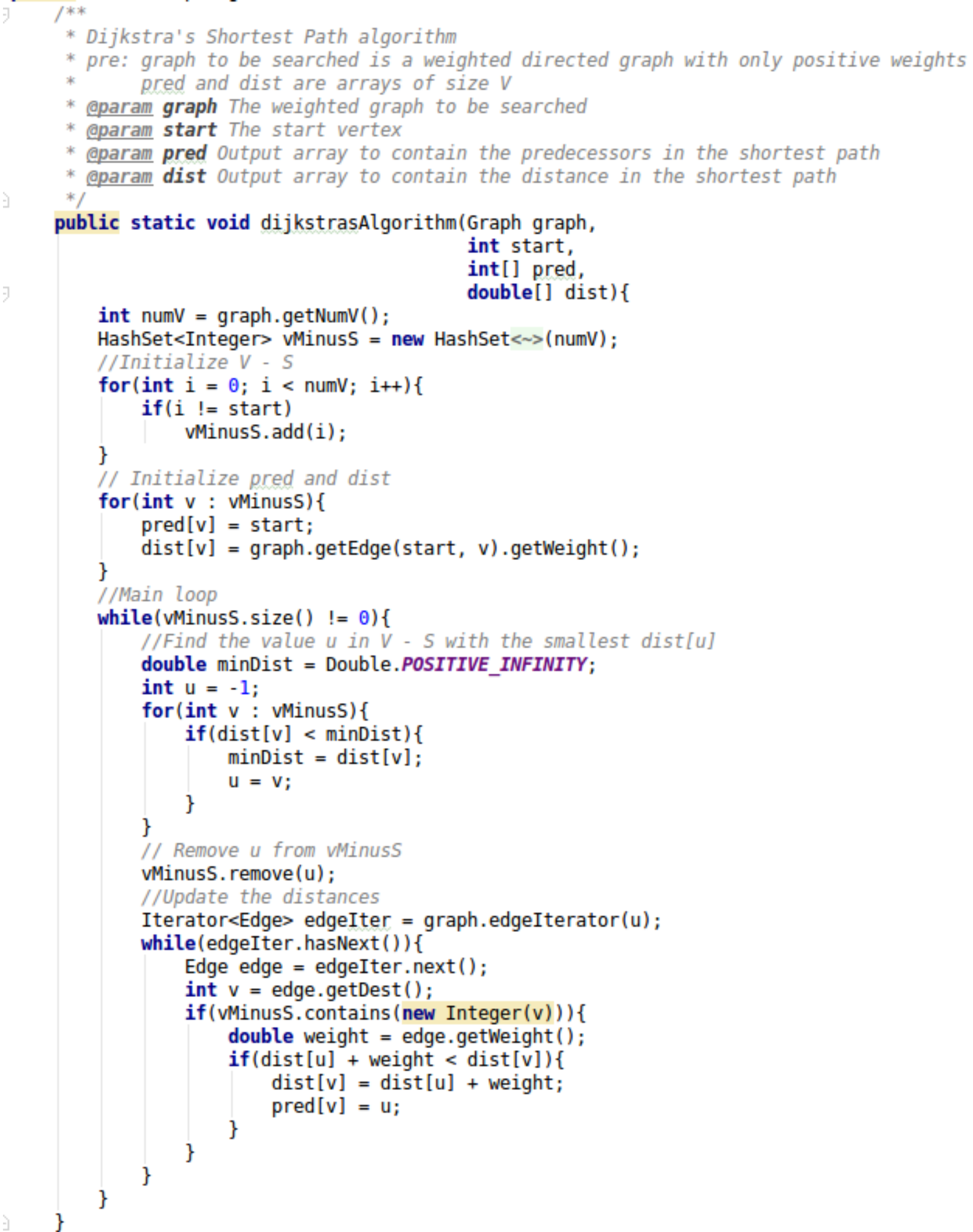
V	d[V]	p[V]	
1	10	0	
2	50	3	★
3	30	0	
4	90	3	

STEP 4



S = {0,1,3,2}
V-S = {4}

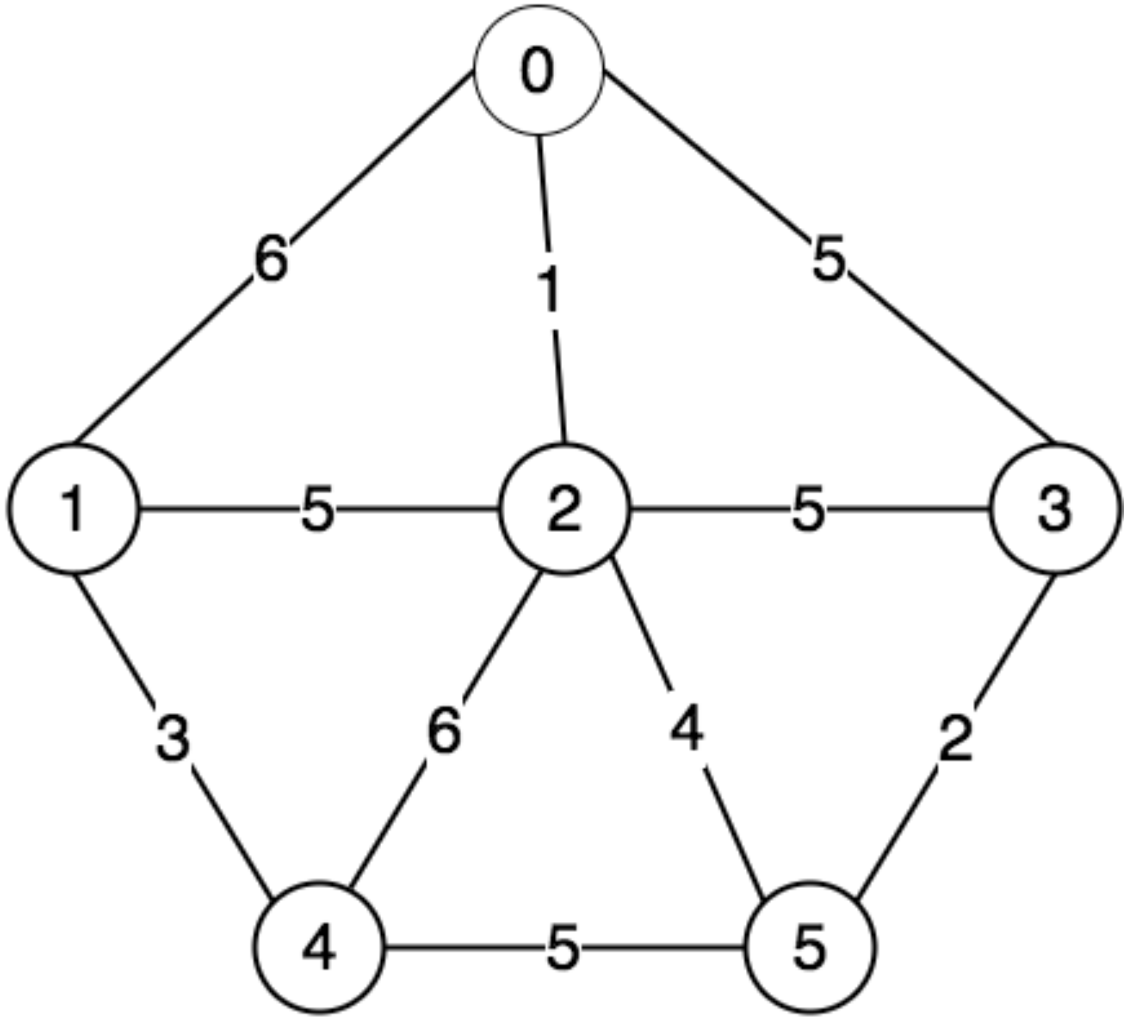
V	d[V]	p[V]
1	10	0
2	50	3
3	30	0
4	60	2



```java
/**
 * Dijkstra's Shortest Path algorithm
 * pre: graph to be searched is a weighted directed graph with only positive weights
 *      pred and dist are arrays of size V
 * @param graph The weighted graph to be searched
 * @param start The start vertex
 * @param pred Output array to contain the predecessors in the shortest path
 * @param dist Output array to contain the distance in the shortest path
 */
public static void dijkstrasAlgorithm(Graph graph,
                                      int start,
                                      int[] pred,
                                      double[] dist){
    int numV = graph.getNumV();
    HashSet<Integer> vMinusS = new HashSet<~>(numV);
    //Initialize V - S
    for(int i = 0; i < numV; i++){
        if(i != start)
            vMinusS.add(i);
    }
    // Initialize pred and dist
    for(int v : vMinusS){
        pred[v] = start;
        dist[v] = graph.getEdge(start, v).getWeight();
    }
    //Main loop
    while(vMinusS.size() != 0){
        //Find the value u in V - S with the smallest dist[u]
        double minDist = Double.POSITIVE_INFINITY;
        int u = -1;
        for(int v : vMinusS){
            if(dist[v] < minDist){
                minDist = dist[v];
                u = v;
            }
        }
        // Remove u from vMinusS
        vMinusS.remove(u);
        //Update the distances
        Iterator<Edge> edgeIter = graph.edgeIterator(u);
        while(edgeIter.hasNext()){
            Edge edge = edgeIter.next();
            int v = edge.getDest();
            if(vMinusS.contains(new Integer(v))){
                double weight = edge.getWeight();
                if(dist[u] + weight < dist[v]){
                    dist[v] = dist[u] + weight;
                    pred[v] = u;
                }
            }
        }
    }
}
```

PRIM

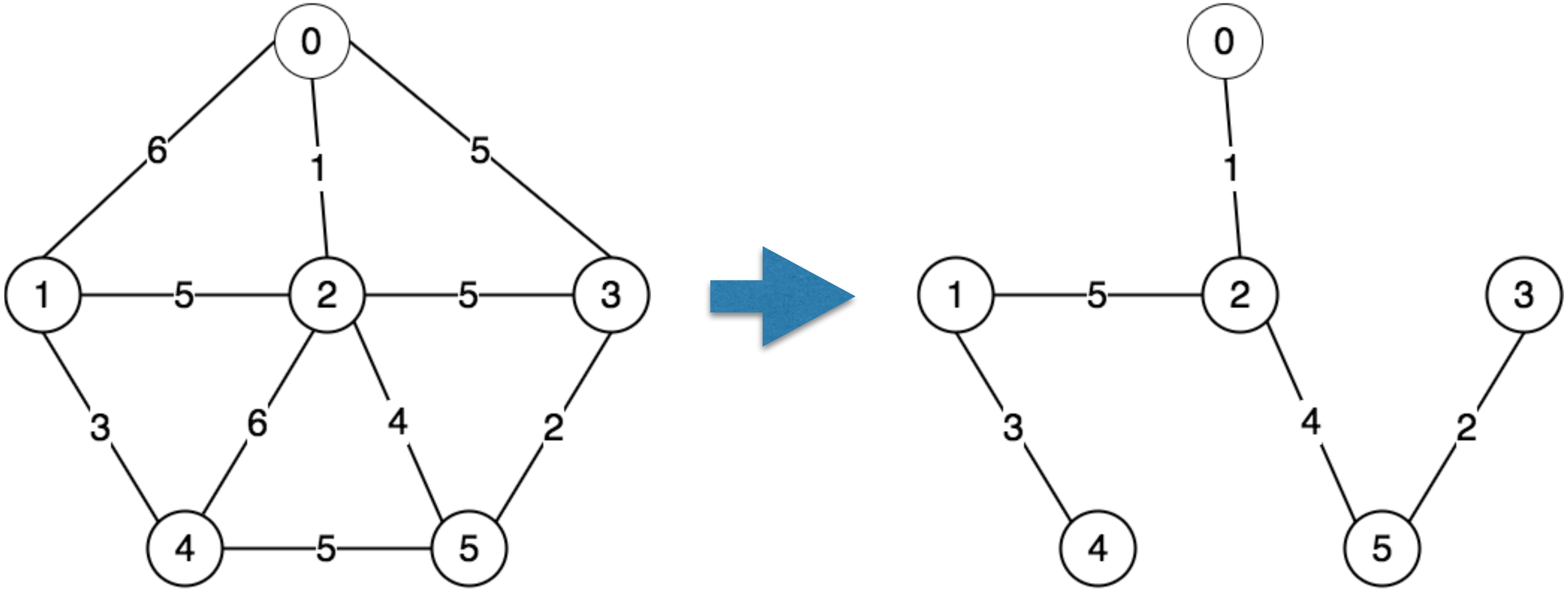
- Find minimum spanning tree in a graph
- Need two sets, S and V-S



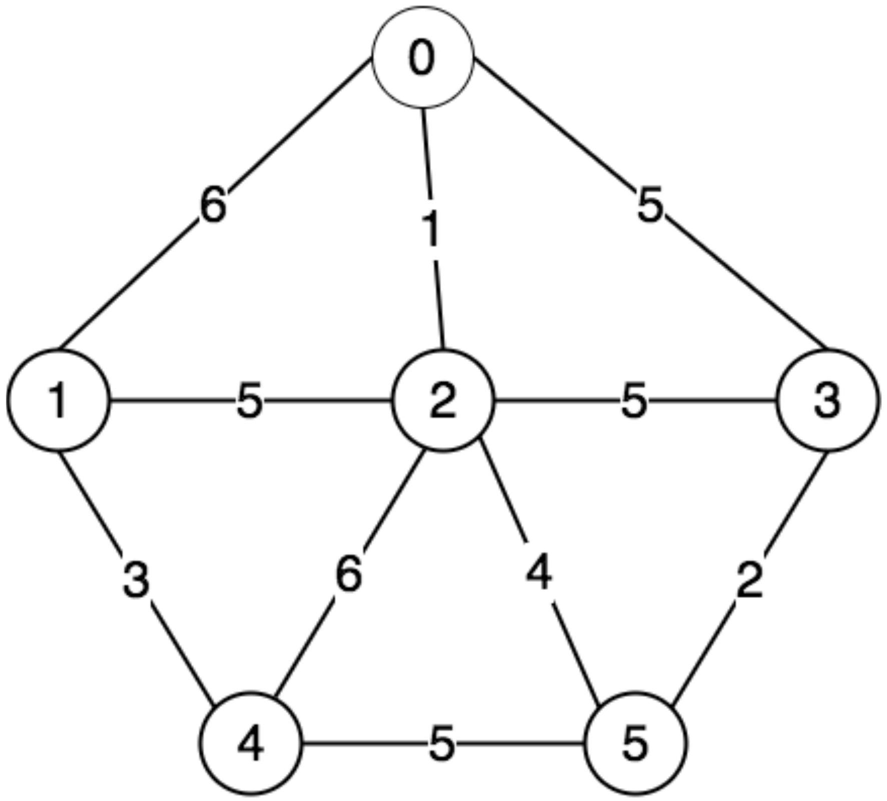
> “For example, if we want to start up our own long-distance phone company and need to connect the cities shown in the given figure, finding minimum spanning tree would allow us to build the cheapest network.”

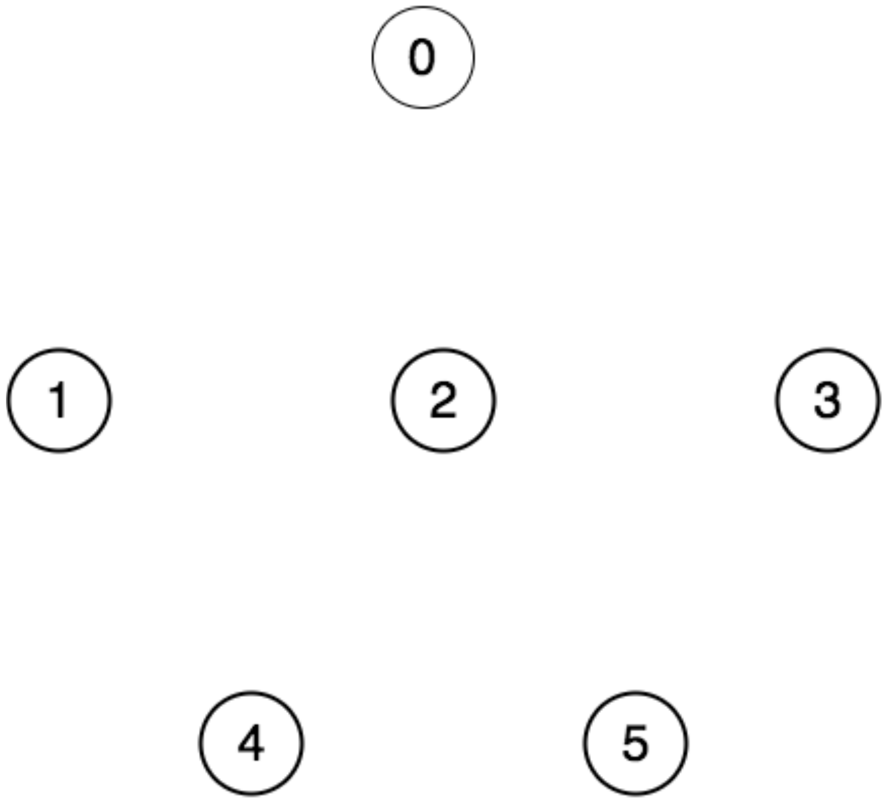
–Elliot & Koffman (the book I stole this example :)

We want this in another term



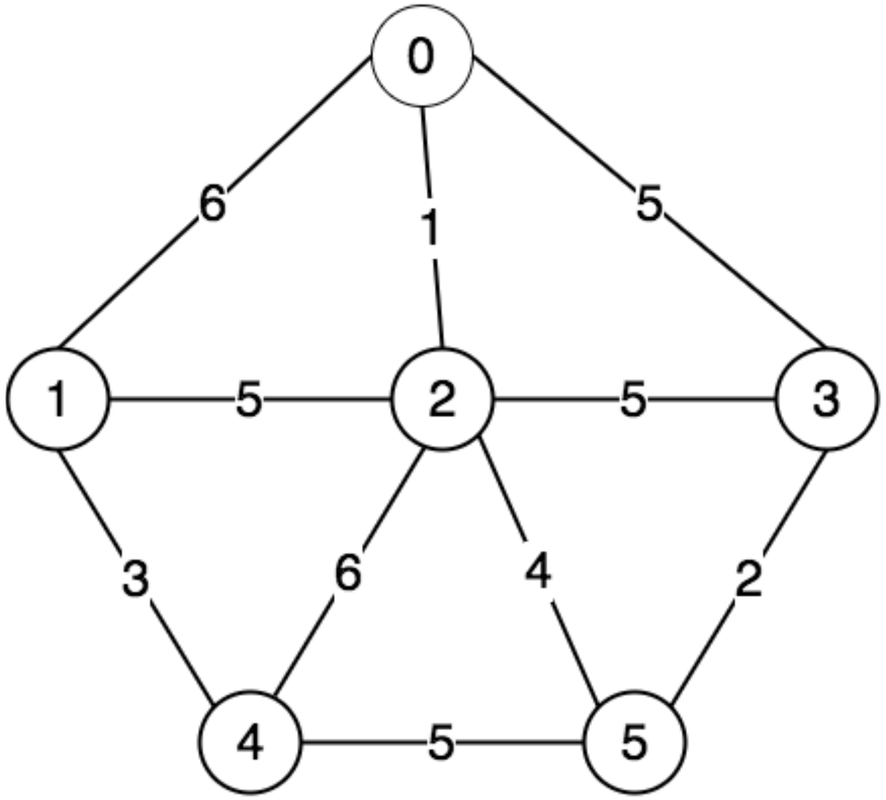
STEP 1

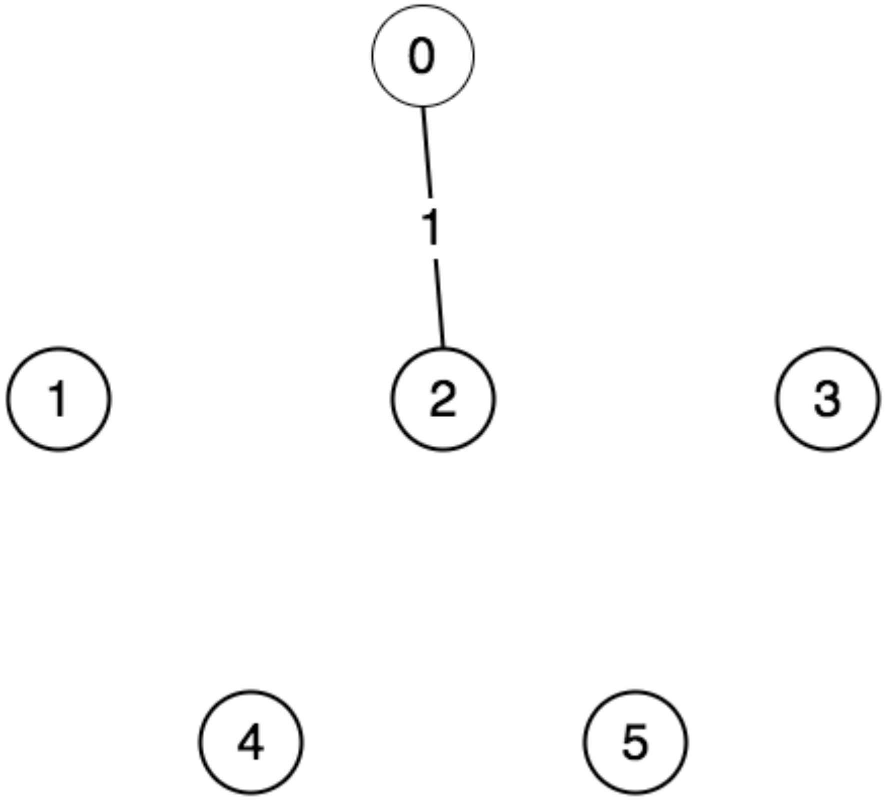




S = {0}
V-S = {1,2,3,4,5}
choose smallest u to v, u from S, v from V-S
smallest choice is (0,2)

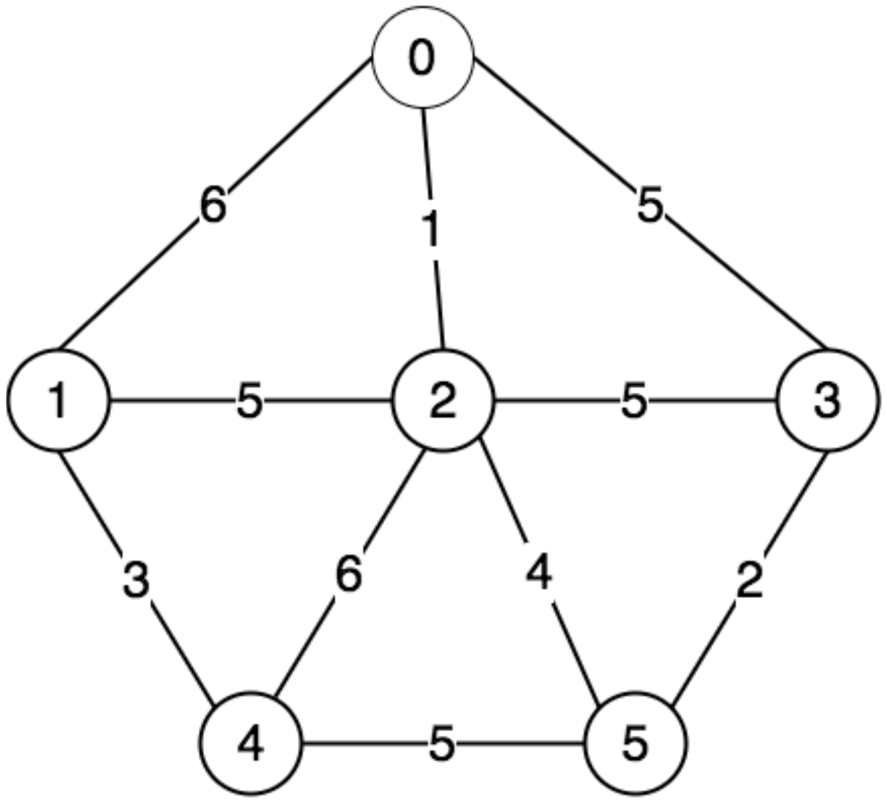
STEP 2

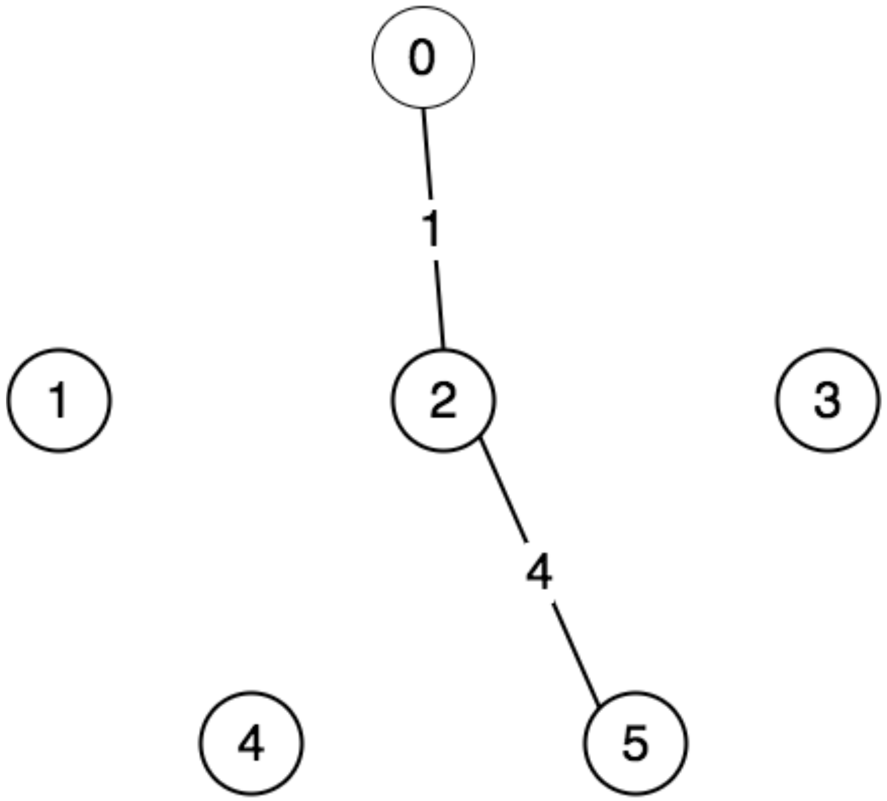




S = {0,2}
V-S = {1,3,4,5}
choose smallest u to v, u from S, v from V-S
smallest choice is (2,5)

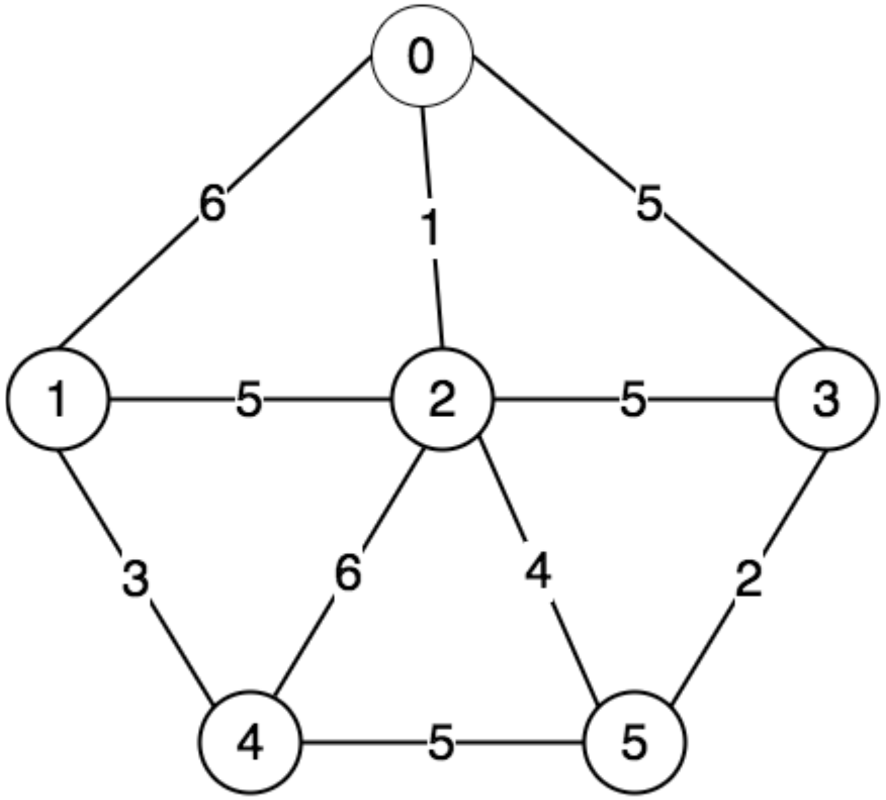
STEP 3

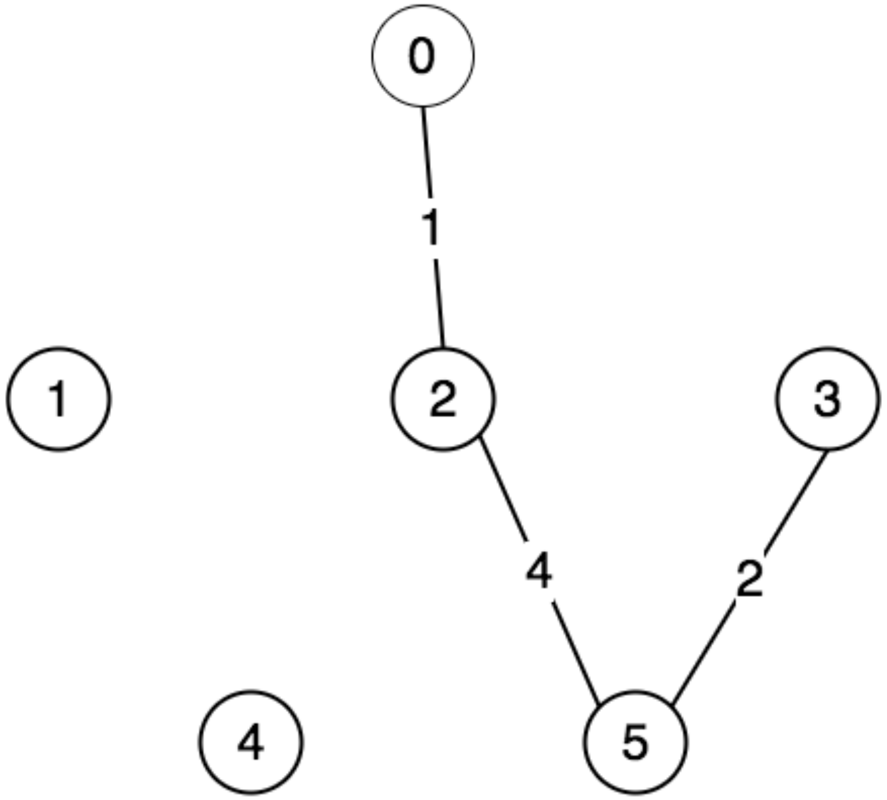




S = {0,2,5}
V-S = {1,3,4}
choose smallest u to v, u from S, v from V-S
smallest choice is (5,3)

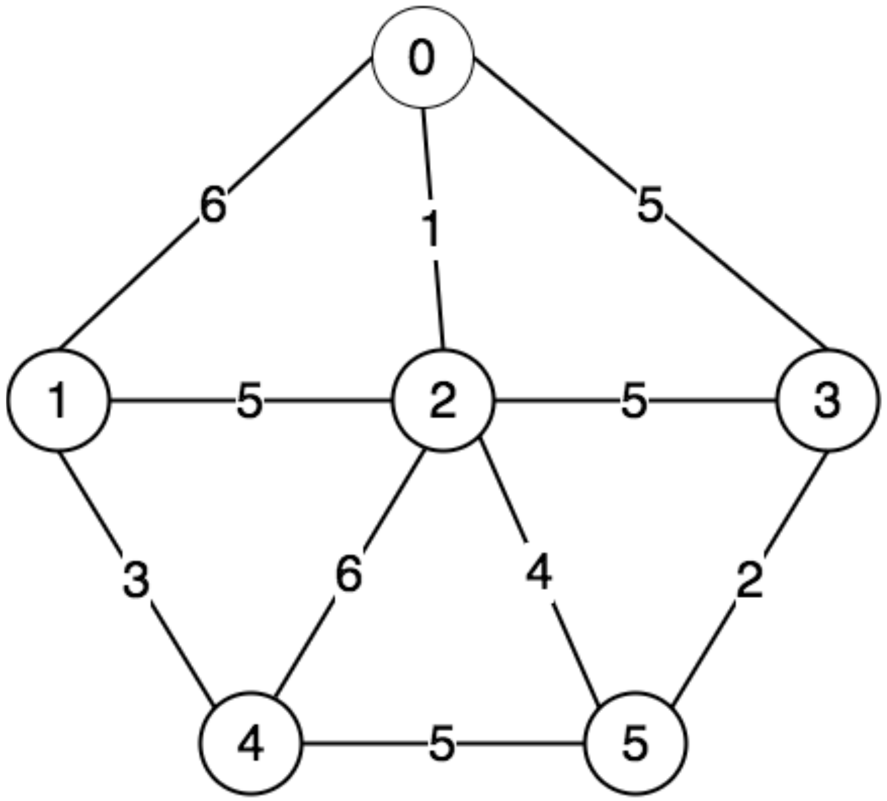
STEP 4

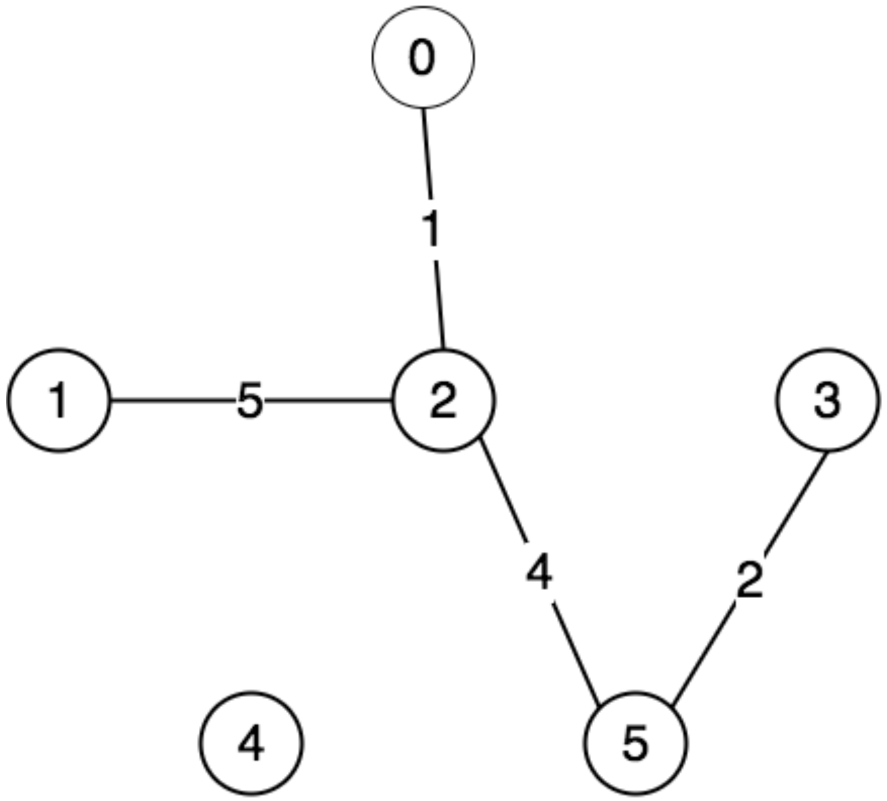




S = {0,2,3,5}
V-S = {1,4}
choose smallest u to v, u from S, v from V-S
smallest choice is (2,1)

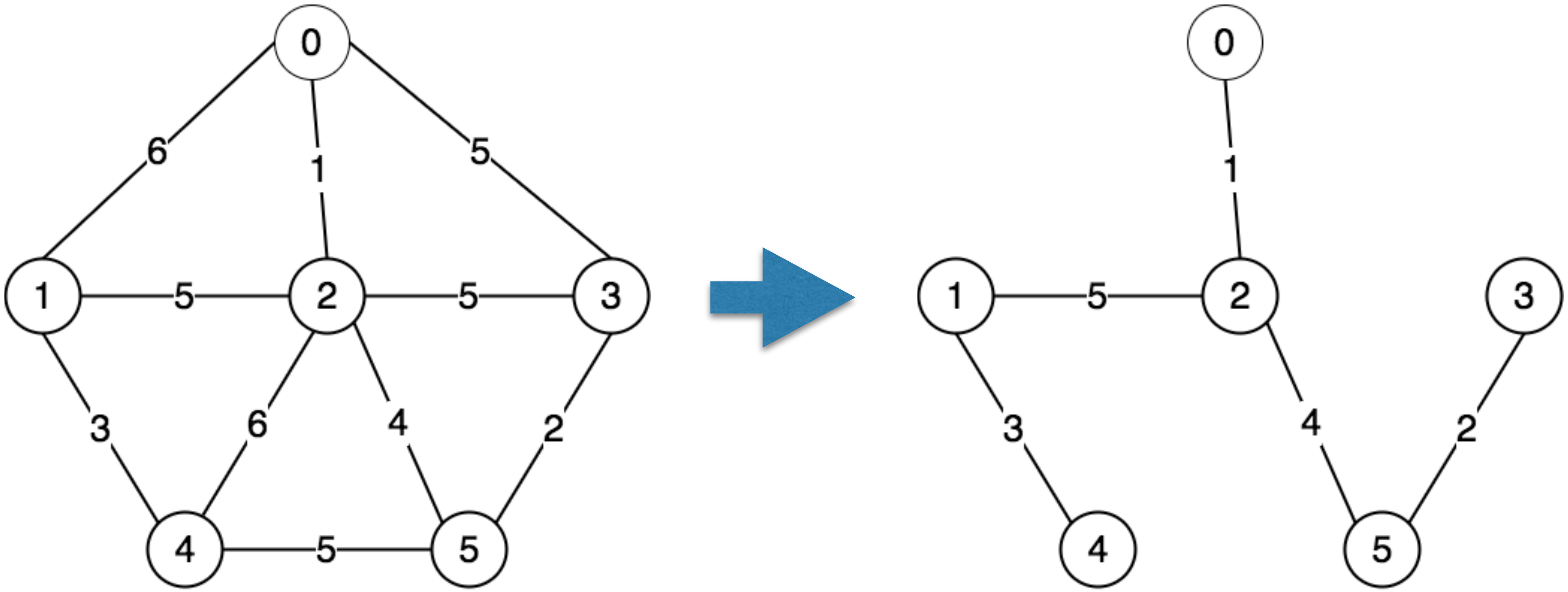
STEP 5

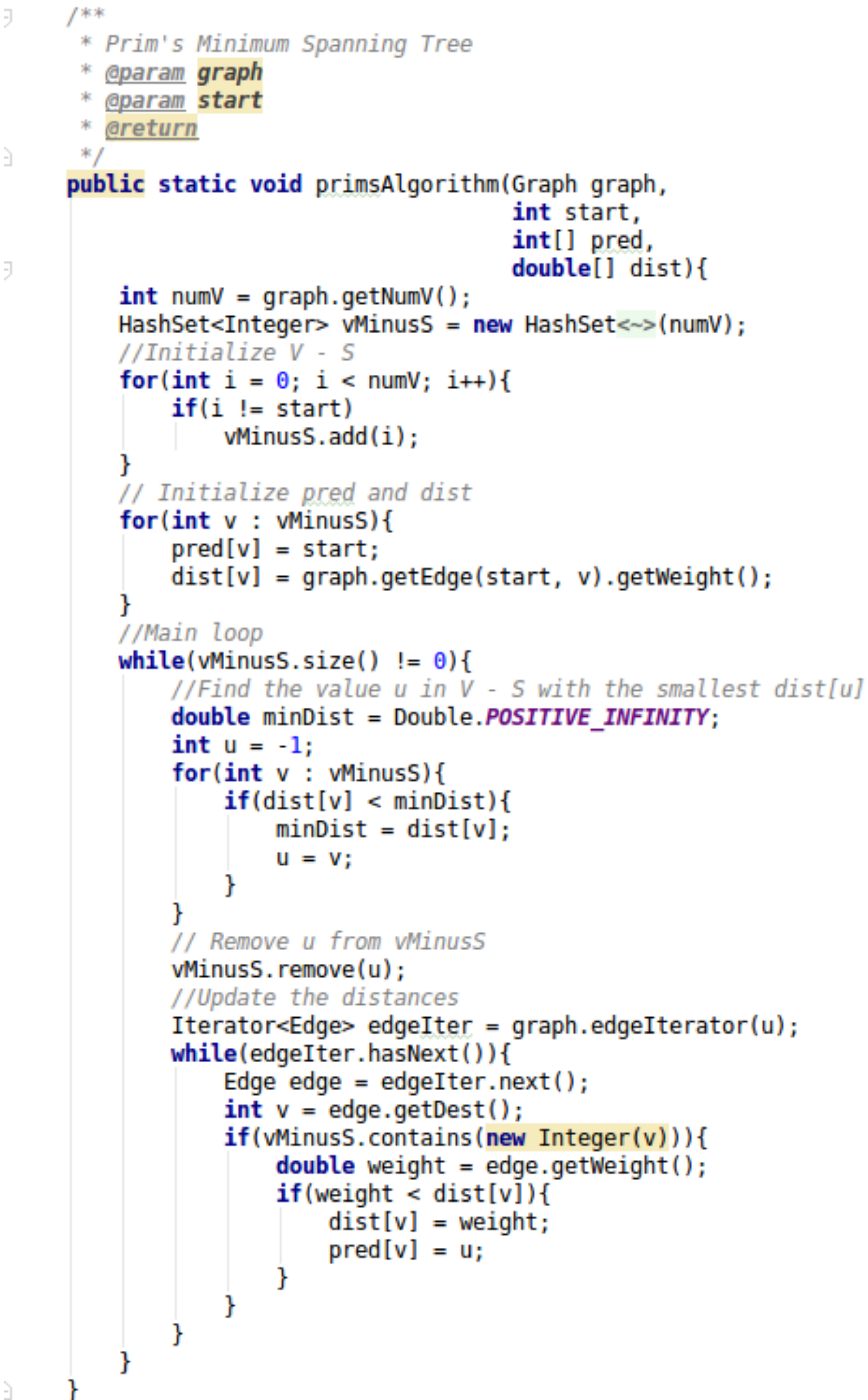




S = {0,1,2,3,5}
V-S = {4}
choose smallest u to v, u from S, v from V-S
smallest choice is (1,4)

STEP 6





```
/**
 * Prim's Minimum Spanning Tree
 * @param graph
 * @param start
 * @return
 */
public static void primsAlgorithm(Graph graph,
                                  int start,
                                  int[] pred,
                                  double[] dist){
    int numV = graph.getNumV();
    HashSet<Integer> vMinusS = new HashSet<~>(numV);
    //Initialize V - S
    for(int i = 0; i < numV; i++){
        if(i != start)
            vMinusS.add(i);
    }
    // Initialize pred and dist
    for(int v : vMinusS){
        pred[v] = start;
        dist[v] = graph.getEdge(start, v).getWeight();
    }
    //Main loop
    while(vMinusS.size() != 0){
        //Find the value u in V - S with the smallest dist[u]
        double minDist = Double.POSITIVE_INFINITY;
        int u = -1;
        for(int v : vMinusS){
            if(dist[v] < minDist){
                minDist = dist[v];
                u = v;
            }
        }
        // Remove u from vMinusS
        vMinusS.remove(u);
        //Update the distances
        Iterator<Edge> edgeIter = graph.edgeIterator(u);
        while(edgeIter.hasNext()){
            Edge edge = edgeIter.next();
            int v = edge.getDest();
            if(vMinusS.contains(new Integer(v))){
                double weight = edge.getWeight();
                if(weight < dist[v]){
                    dist[v] = weight;
                    pred[v] = u;
                }
            }
        }
    }
}
```

рахмат
danke
謝謝
ngiyabonga
teşekkür ederim
Баярлалаа
spas
mersi
kia ora
barka
welalin
tack
mahalo
tapadh leat
спасибо
faafetai lava
vinaka
спасибі
blagodaram
dank je
misaotra
paldies
grazzi
matondo
kiitos
dankie
nanni
nandri
dhanyavad
mauruuru
köszönöm
thank you
gracias
хвала
asante
manana
bedankt
bayarlalaa
gracie
hvala
dankon
aciū
akun
obrigada
tenki
murakoze
enkosi
djiere dieuf
tau
mochchakkeram
dziękuję
дякую
mamnun
chokrane
chnorakaloutioun
gratias ago
gràcies
sulpay
go raibh maith agat
sobodi
dėkuji
sagolun
sukriya
kop khun krap
taiku
obrigado
arigatô
takk
dakujem
trugarez
mèsi
didi madloba
ありがとう
tanemirt
grazie
najis tuke
kam sah hamnida
rahmat
terima kasih
rahmet
ευχαριστώ
diolch
dhanyavadagalu
shukriya
merce
мерси
তোমাকে ধন্যবাদ
감사합니다
xièxie
merci

References

- https://en.wikipedia.org/wiki/Dijkstras_algorithm
- https://en.wikipedia.org/wiki/Prim%27s_algorithm
- https://github.com/jimlay14/Data-Structures
- http://www.vogella.com/tutorials/JavaAlgorithmsDijkstra/article.html
- Koffman & Wolfgang; Objects, Abstraction, Data Structures and Design using JAVA [book]